Preliminary Interface Specifications

ROD Commands and Registers

A possible framework for the RCC/ROD interface is described in this document. Some of the framework's goals are:

clearly define the RCC/ROD interface
provide the RCC with both low-level and high-level commands
relieve the RCC of burdensome real-time constraints
relieve the RCC of data moving and processing burdens
prohibit the RCC from affecting the ROD in inappropriate ways
allow the RCC direct access to ROD resources, primarily its various memory spaces, for debugging purposes
enhance independent development and debugging of RCC software and the ROD and its software

The ROD/RCC interface has four parts:
the Host Port Interface (HPI)
the command set
the register set
the block-transfer FIFO (BLT FIFO)

The HPI is mainly used immediately after power-up for downloading DSP code, but may also contain ports dedicated to hardware reset, RODBUSY monitoring, and debugging.

The register set and command set work together to implement the bulk of the RCC/ROD interface. Both are implemented in the ROD's VME-accessible dual-port RAM. Registers are simply locations in dual-port RAM that have pre-defined meanings. Hardware protects each register from simultaneous access by both the ROD and RCC.

Registers come in several flavors. Some registers, such as State and L1ID, give the RCC immediate access to the state of ROD. The ROD may update them at any time, and the RCC may read them at any time. Other registers are only updated in response to a command from the RCC. Some error or status registers may fall into this class. By requiring a command to update these registers, the ROD can insure that the RCC sees a consistent set of registers.

A small set of registers and the BLT FIFO are used to implement the command interface. The RCC acts as master. It issues a command by writing it parameters to the appropriate registers and then writing the command word to the Command register. The ROD is the slave and executes the command. Any command results are returned in the dual-port RAM or, optionally, the BLT FIFO. Upon completing the command, the ROD places the command status (normally stOK) in CommandStatus and Null in Command and optionally interrupts the RCC. The RCC may abort a command in progress by writing a nonzero value to the Abort register. In response, the ROD aborts the command and places stABORTED in CommandStatus and Null in Command.

It is unreasonable to force every RCC/ROD interaction into the same command model. For example, trigger rate and capture criteria influence how much time is required to capture an event. It would be too restrictive to prohibit the RCC from issuing new commands to the ROD while the capture is pending. The dedicated CaptureMode and CaptureStatus registers allow for an efficient event capture mechanism. A similar mechanism, not reflected in the tables below, may prove useful for histogram readout.

The command word layout is shown below. Dedicated bits are used to tell the ROD:
if it should interrupt the RCC when the command completes,
if it should return results in the VME BLT FIFO rather than VME dual-port RAM,
the set of modules the command should be applied to.

Command Word Layout

```
Command =   IBrr rrrr rrrr CCCC   CCCC CCCC MMMM MMMM

      I = interrupt RCC upon command completion
      B = return results in via VME BLT FIFO rather than VME dual-port RAM
      r = reserved for future use
      C = command opcode
      M = module specifier:
        0-95 = apply command to the specified module
        0x8S = apply command to all modules in set S (module sets are contained in DPRAM)
        0xf0 = apply command to no modules
        0xff = apply command to all modules
```

Registers in the ROD's VME Dual-Ported RAM

Each register shall be documented with:

- Name and description of the register
- Read/Write expectations for ROD and RCC
- List of ROD states in which the register may be modified

Register Name	**Words**	**RCC**	**ROD**	**Description**
State	1	R	W	state of the ROD: IDLE, Running, etc.
L1ID	1	R	W	the L1 ID information for the event most recently sent to the S-LINK FIFO
L1Count	1	R	W	a count of the number of L1's processed this run
CaptureMode	1	W	R	flags indicating such things as whether the RCC should be interrupted upon capture completion, etc.
CaptureStatus	1	R	W	indication of whether captures are complete, pending, etc.
ModuleSet0	3	W	R	see Command Word Layout
ModuleSet1	3	W	R	see Command Word Layout
ModuleSet2	3	W	R	see Command Word Layout
ModuleSet3	3	W	R	see Command Word Layout
Abort	1	W	R/W	To abort a command in progress, the RCC writes a nonzero value to Abort.
CommandStatus	1	R	W	written by ROD upon completion of each command
Command	1	W	R/W	The RCC writes a command to Command. The ROD polls Command and, if it is not Null, executes the command. Upon completing the command, ROD places Null in Command and the command status in CommandStatus.
TBD<command parameters>	TBD	W/R	R	parameters for commands
TBD<command return values>	TBD	R	W	values returned by commands
TBD<error counts/flags>				
TBD<debug registers>				

Some Commands in the ROD Command Set

Each command shall be documented with:

- Name and description of the command
- Name and description of each parameter
- Name and description of each value returned
- Legal values of the I, B and M bits in the command word
- List of ROD states in which the ROD would expect to receive the command
- List of conditions that may cause the command to return an error
- List of conditions that must be met before the ROD will execute the command

Command Name	**Parameters**	**Returns**	
Null	N/A	N/A	Null is a placeholder only
Initialize	<none>	<nothing>	initialize the ROD
StartPhysicsRun	<none>	<nothing>	enter physics run state
FinishPhysicsRun	<none>	<nothing>	leave physics run state once all events have been transmitted
AbortPhysicsRun	<none>	<nothing>	leave physics run state immediately
WriteMemory	DSP#, address, data count, data	<nothing>	write directly to DSP memory
ReadMemory	DSP#, address, data count	data from DSP memory	read directly from DSP memory
WriteConfiguration	data count, data words	<nothing>	place configuration data into ROD memory
ConfigureModule	<none>	<nothing>	configure module using configuration image in ROD memory
ReadConfigurationImage	<none>	configuration image	read configuration image from ROD memory
FetchRegisterData	<none>	<nothing>	get register data from module(s) and place in ROD memory
ReadRegisterDataImage	<none>	register data image	read register data image from ROD memory
ClearErrorRegister	<none>	<nothing>	zero specified error register
ClearAllErrorRegisters	<none>	<nothing>	zero all error registers
SetCaptureCriteria	event type, size, number of events to capture, etc.	<nothing>	set criteria for event(s) to be captured
ReadCaptureImage	<none>	event(s) size, event data	read event image from ROD memory
ReadCaptureLeaderTrailer	<none>	event leader and trailer (no data elements)	read event leader and trailer from image in ROD memory
NextCapture	<none>	<nothing>	arm next capture
ReadHistogram	histogram ID	histogram size, histogram	read a current histogram from ROD memory (using histogram paging method)
ReduceHistogram	histogram ID	error count, etc.	reduce a specified histogram, e.g., by calculating sum and sum of squares, place results in ROD memory (probably not used at run time)
ReadReductionImage	<none>	size, reduction image	read a reduced histogram from ROD memory
ClearHistogram	histogram ID	<nothing>	zero a histogram (probably not used at run time)
SetDAC	DAC value	<nothing>	set a front-end DAC
Calibrate	pulse count	error count, etc.	issue calibrate and L1 commands to the specified modules, collect and/or histogram the resulting data
Scan	pulse count, scanpoint count, initial DAC value, DAC step	error count, etc.	perform calibration scan, histogram the resulting data

The word “image” refers to information stored in ROD memory, as opposed to information the ROD may need to acquire from modules or ROD hardware.

Sparsifier/ASM II Interface

I. The Sparsifier transmits a clock and 17 parallel control bits to the ASM II via optical G-Link.

Clock: ~40 MHz (LHC BC clock)

Control Bits:

Name	Count	Description	comment
WA0-7	8	SCA Write Address	
WR_CLK	1	SCA Write Clock (20 MHz)	
RD_CLK	1	SCA Read Clock = (6.67 MHz)	
GA0-1	2	SCA Channel Select	
RD	1	SCA Read address strobe	
SD	1	SCA Serial read address	
TX_DAV	1	per-word enable for G-Link transmitter	simplifies ASM/Sparsifier synchronization
CAL	1	Calibration strobe	
DAC_CLK	1	clock for serial data into Calibration DAC	
DAC_D	*0	serial data for Calibration DAC	same as SD
ADC_ENCODE	*0	start conversion (ADC clock)	derived from RDCLK
SR_LOAD	*0	load ADC value into output shift register	derived from RDCLK
Total	**17**		

* If the 1022/1024 G-Link chip set is used, these signals should be implemented as dedicated control bits instead of being derived from other control bits. The older 1022/1024 G-Link chip set can transmit up to 21 bits plus clock. The newer 1032/1034 G-Link chip set uses less power than the older chip set but can transmit at most 17 bits plus clock.

II. The ASM II transmits ADC data to the Sparsifier via two optical G-Links operating at the beam clock rate.

Only digitized SCA samples are routinely transmitted. For example, data from ADC's dedicated to monitoring voltages or temperature might be transmitted during a system-wide reset period using G-Link bits that are normally dedicated to transmitting SCA samples.

The G-Link word size is 16 bits, for a total of 32 bits per ASM II. Two bits are dedicated to each of the 16 main ADC's on the ASM II. There are no framing bits, CRC bits, etc. There is no means of detecting bit errors in the data. Synchronization is established via the TX_DAV mechanism as shown in the "Sparsifier-ASM Interaction" diagram on the next page.

III. If required by the ASM II design, the Sparsifier helps the ASM II establish/reestablish G-Link lock.

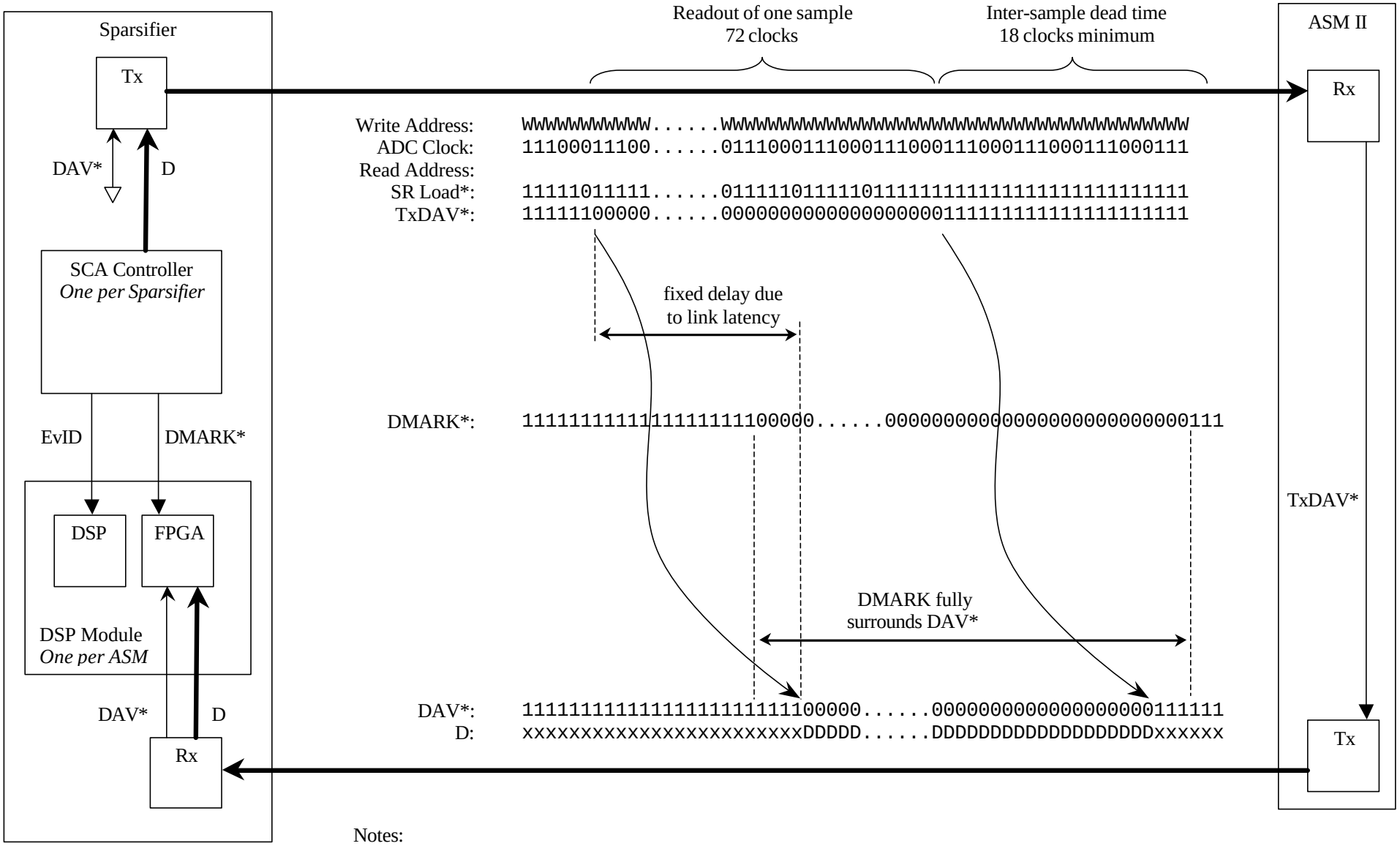
The G-Link receiver on the ASM II needs a reference frequency to establish lock. This reference frequency can come from a local source on the ASM, such as a crystal oscillator, or from the Sparsifier via the optical fiber. If there is no local crystal oscillator on the ASM II, then the Sparsifier must transmit an Idle Word to an ASM II if it sees that both data links for the ASM II are out of lock. The Idle Word's bit pattern creates a clock at the word rate. The ASM II must provide a mechanism to use this clock as the reference clock until frequency lock is obtained. The 1024 G-Link receiver has this mechanism built in. The 1034 does not.

IV. Laser Safety.

If laser safety is determined to be a concern, the Sparsifier and ASM II will have the responsibilities listed below, which ensure that lasers are disabled if an entire optical cable is severed. For example, all the optical fibers for a single chamber may be carried in a single cable.

The ASM II must disable its lasers whenever its clock/control link is out of lock. This rule can be implemented with little or no additional circuitry on the ASM II and assumes that data and clock/control fibers for an ASM are in the same cable.

The Sparsifier must disable all of the lasers it drives into a cable under certain conditions (to be determined). For example, if several data links within a single cable cannot retain or establish lock, then the Sparsifier must disable all lasers it drives into that cable.



Notes:

1. EvID is a serial event identifier that contains L1#, BC#, SCA read addresses, first sample location, etc.. The first sample location tells the DSP which sample in its memory is the first sample for the event.
2. DMARK* is a signal dedicated to notifying the FPGA when it should expect to see data arriving on the G-Link. It need not be precisely timed, because the G-Link Rx outputs a DAV* signal to the FPGA.

ROD/Peripherals Interface

Peripherals include:

RCC	--	ROD Crate Controller
TIM	--	Timing Interface Module
ROL	--	Readout Link
ROB	--	Readout Buffer
DCS	--	Detector Control System

All of these devices or systems will have ATLAS-standard interface requirements. Below are links to currently available standard interface information.

General Trigger/DAQ Interface to Front End systems:
http://atlasinfo.cern.ch/Atlas/GROUPS/DAQTRIG/DIG/document/FEdoc_2.5.pdf

TIM:
http://www.hep.ucl.ac.uk/~jbl/SCT/TIM_welcome.html

S-LINK (used for ROL):
http://hsi.web.cern.ch/HSI/s-link/

Event Format at ROB Input:
http://atddoc.cern.ch/Atlas/Notes/050/Note050-1.html

DCS:
http://atlasinfo.cern.ch/ATLAS/GROUPS/DAQTRIG/DCS/dcshome.html

ROD Crate:
http://atlas.web.cern.ch/Atlas/GROUPS/FRONTEND/documents/crate_od.pdf

RCC: